A Discriminative Global Training Algorithm for Statistical MT

Christoph Tillmann
IBM T.J. Watson Research Center
Yorktown Heights, N.Y. 10598
ctill@us.ibm.com

Tong Zhang
Yahoo! Research
New York City, N.Y. 10011
tzhang@yahoo-inc.com

Abstract

This paper presents a novel training algorithm for a linearly-scored block sequence translation model. The key component is a new procedure to directly optimize the global scoring function used by a SMT decoder. No translation, language, or distortion model probabilities are used as in earlier work on SMT. Therefore our method, which employs less domain specific knowledge, is both simpler and more extensible than previous approaches. Moreover, the training procedure treats the decoder as a black-box, and thus can be used to optimize any decoding scheme. The training algorithm is evaluated on a standard Arabic-English translation task.

1 Introduction

This paper presents a view of phrase-based SMT as a sequential process that generates block orientation sequences. A block is a pair of phrases which are translations of each other. For example, Figure 1 shows an Arabic-English translation example that uses four blocks. During decoding, we view translation as a block segmentation process, where the input sentence is segmented from left to right and the target sentence is generated from bottom to top, one block at a time. A monotone block sequence is generated except for the possibility to handle some local phrase re-ordering. In this local re-ordering model (Tillmann and Zhang, 2005; Kumar and Byrne, 2005) a block b with orientation o is generated relative to its predecessor block b'. During decoding, we maximize the score $s_w(b_1^n, o_1^n)$ of a block orientation sequence

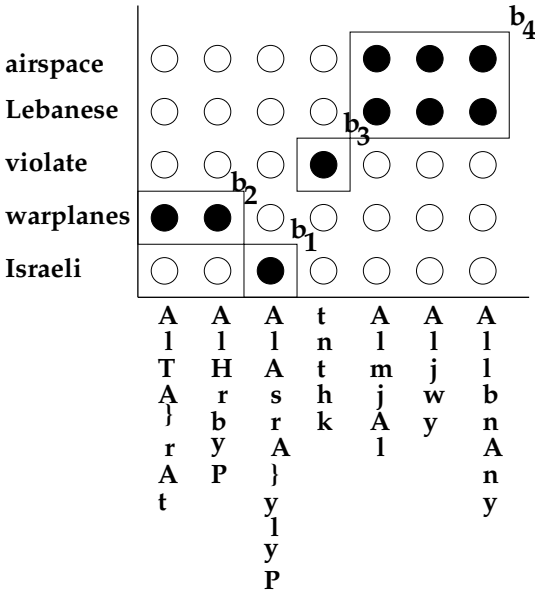

Figure 1: An Arabic-English block translation example, where the Arabic words are romanized. The following orientation sequence is generated: $o_1 = N, o_2 = L, o_3 = N, o_4 = R$.

(b_1^n, o_1^n):

$$s_w(b_1^n, o_1^n) = \sum_{i=1}^{n} w^T \cdot f(b_i, o_i, b_{i-1}), \quad (1)$$

where b_i is a block, b_{i-1} is its predecessor block, and $o_i \in \{L(\text{eft}), R(\text{ight}), N(\text{eutral})\}$ is a three-valued orientation component linked to the block b_i: a block is generated to the left or the right of its predecessor block b_{i-1}, where the orientation o_{i-1} of the predecessor block is ignored. Here, n is the number of blocks in the translation. We are interested in learning the weight vector w from the training data. $f(b_i, o_i, b_{i-1})$ is a high-dimensional binary feature representation of the block orientation pair (b_i, o_i, b_{i-1}). The block orientation se-

quence is generated under the restriction that the concatenated source phrases of the blocks b_i yield the input sentence. In modeling a block sequence, we emphasize adjacent block neighbors that have right or left orientation, since in the current experiments only local block swapping is handled (neutral orientation is used for 'detached' blocks as described in (Tillmann and Zhang, 2005)).

This paper focuses on the discriminative training of the weight vector w used in Eq. 1. The decoding process is decomposed into local decision steps based on Eq. 1, but the model is trained in a global setting as shown below. The advantage of this approach is that it can easily handle tens of millions of features, e.g. up to 35 million features for the experiments in this paper. Moreover, under this view, SMT becomes quite similar to sequential natural language annotation problems such as part-of-speech tagging and shallow parsing, and the novel training algorithm presented in this paper is actually most similar to work on training algorithms presented for these task. e.g. the on-line training algorithm presented in (McDonald et al., 2005) and the perceptron training algorithm presented in (Collins, 2002). The current approach does not use specialized probability features as in (Och, 2003) in any stage during decoder parameter training. Such probability features include language model, translation or distortion probabilities, which are commonly used in current SMT approaches [1]. We are able to achieve comparable performance to (Tillmann and Zhang, 2005). The novel algorithm differs computationally from earlier work in discriminative training algorithms for SMT (Och, 2003) as follows:

- No computationally expensive N-best lists are generated during training: for each input sentence a single block sequence is generated on each iteration over the training data.

- No additional development data set is necessary as the weight vector w is trained on bilingual training data only.

The paper is structured as follows: Section 2 presents the baseline block sequence model and the feature representation. Section 3 presents the discriminative training algorithm that learns a good global ranking function used during decoding. Section 4 presents results on a standard Arabic-English translation task. Finally, some discussion and future work is presented in Section 5.

2 Block Sequence Model

This paper views phrase-based SMT as a block sequence generation process. Blocks are phrase pairs consisting of target and source phrases and local phrase re-ordering is handled by including so-called block orientation. Starting point for the block-based translation model is a block set, e.g. about 9.5 million Arabic-English phrase pairs for the experiments in this paper. This block set is used to decode training sentence to obtain block orientation sequences that are used in the discriminative parameter training. Nothing but the block set and the parallel training data is used to carry out the training. We use the block set described in (Al-Onaizan et al., 2004), the use of a different block set may effect translation results.
Rather than predicting local block neighbors as in (Tillmann and Zhang, 2005) , here the model parameters are trained in a global setting. Starting with a simple model, the training data is decoded multiple times: the weight vector w is trained to discriminate block sequences with a high translation score against block sequences with a high BLEU score [2]. The high BLEU scoring block sequences are obtained as follows: the regular phrase-based decoder is modified in a way that it uses the BLEU score as optimization criterion (independent of any translation model). Here, searching for the highest BLEU scoring block sequence is restricted to local re-ordering as is the model-based decoding (as shown in Fig. 1). The BLEU score is computed with respect to the single reference translation provided by the parallel training data. A block sequence with an average BLEU score of about 0.54 is obtained for each training sentence [3]. The 'true' maximum BLEU block sequence as well as the high scoring

[1] A translation and distortion model is used in generating the block set used in the experiments, but these translation probabilities are not used during decoding.

[2] High scoring block sequences may contain translation errors that are quantified by a lower BLEU score.

[3] The training BLEU score is computed for each training sentence pair separately (treating each sentence pair as a single-sentence corpus with a single reference) and then averaged over all training sentences. Although block sequences are found with a high BLEU score on average there is no guarantee to find the maximum BLEU block sequence for a given sentence pair. The target word sequence corresponding to a block sequence does not have to match the reference translation, i.e. maximum BLEU scores are quite low for some training sentences.

block sequences are represented by high dimensional feature vectors using the binary features defined below and the translation process is handled as a multi-class classification problem in which each block sequence represents a possible class. The effect of this training procedure can be seen in Figure 2: each decoding step on the training data adds a high-scoring block sequence to the discriminative training and decoding performance on the training data is improved after each iteration (along with the test data decoding performance). A theoretical justification for the novel training procedure is given in Section 3.

We now define the feature components for the block bigram feature vector $\mathbf{f}(b_i, o_i, b_{i-1})$ in Eq. 1. Although the training algorithm can handle real-valued features as used in (Och, 2003; Tillmann and Zhang, 2005) the current paper intentionally excludes them. The current feature functions are similar to those used in common phrase-based translation systems: for them it has been shown that good translation performance can be achieved [4]. A systematic analysis of the novel training algorithm will allow us to include much more sophisticated features in future experiments, i.e. POS-based features, syntactic or hierarchical features (Chiang, 2005). The dimensionality of the feature vector $\mathbf{f}(b_i, o_i, b_{i-1})$ depends on the number of binary features. For illustration purposes, the binary features are chosen such that they yield 1 on the example block sequence in Fig. 1. There are **phrase-based** and **word-based** features:

$$f_{1000}(b_i, o_i, b_{i-1}) =$$
$$= \begin{cases} 1 & \text{block } b_i \text{ consists of target phrase} \\ & \text{'violate' and source phrase 'tnthk'} \\ 0 & \text{otherwise} \end{cases}$$
$$f_{1001}(b_i, o_i, b_{i-1}) =$$
$$= \begin{cases} 1 & \text{'Lebanese' is a word in the target} \\ & \text{phrase of block } b_i \text{ and 'AllbnAny'} \\ & \text{is a word in the source phrase} \\ 0 & \text{otherwise} \end{cases}$$

The feature f_{1000} is a 'unigram' phrase-based feature capturing the identity of a block. Additional phrase-based features include block orientation, target and source phrase bigram features. Word-based features are used as well, e.g. feature f_{1001} captures word-to-word translation dependencies similar to the use of Model 1 probabilities in (Koehn et al., 2003). Additionally, we use distortion features involving relative source word position and m-gram features for adjacent target words. These features correspond to the use of a language model, but the weights for theses features are trained on the parallel training data only. For the most complex model, the number of features is about 35 million (ignoring all features that occur only once).

[4] On our test set, (Tillmann and Zhang, 2005) reports a BLEU score of 37.8 and (Ittycheriah and Roukos, 2005) reports a BLEU score of 46.0 using real-valued features only.

3 Approximate Relevant Set Method

Throughout the section, we let $\mathbf{z} = (b_1^n, o_1^n)$. Each block sequence $\mathbf{z} = (b_1^n, o_1^n)$ corresponds to a candidate translation. In the training data where target translations are given, a BLEU score $\mathrm{Bl}(\mathbf{z})$ can be calculated for each $\mathbf{z} = (b_1^n, o_1^n)$ against the target translations. In this set up, our goal is to find a weight vector w such that the higher $s_w(\mathbf{z})$ is, the higher the corresponding BLEU score $\mathrm{Bl}(\mathbf{z})$ should be. If we can find such a weight vector, then block decoding by searching for the highest $s_w(\mathbf{z})$ will lead to good translation with high BLEU score.

Formally, we denote a source sentence by $\mathbf{S}$, and let $V(\mathbf{S})$ be the set of possible candidate oriented block sequences $\mathbf{z} = (b_1^n, o_1^n)$ that the decoder can generate from $\mathbf{S}$. For example, in a monotone decoder, the set $V(\mathbf{S})$ contains block sequences $\{b_1^n\}$ that cover the source sentence $\mathbf{S}$ in the same order. For a decoder with local re-ordering, the candidate set $V(\mathbf{S})$ also includes additional block sequences with re-ordered block configurations that the decoder can efficiently search. Therefore depending on the specific implementation of the decoder, the set $V(\mathbf{S})$ can be different. In general, $V(\mathbf{S})$ is a subset of all possible oriented block sequences $\{(b_1^n, o_1^n)\}$ that are consistent with input sentence $\mathbf{S}$.

Given a scoring function $s_w(\cdot)$ and an input sentence $\mathbf{S}$, we can assume that the decoder implements the following decoding rule:

$$\hat{\mathbf{z}}(\mathbf{S}) = \arg \max_{\mathbf{z} \in V(\mathbf{S})} s_w(\mathbf{z}). \qquad (2)$$

Let $\mathbf{S}_1, \ldots, \mathbf{S}_N$ be a set of N training sentences. Each sentence $\mathbf{S}_i$ is associated with a set $V(\mathbf{S}_i)$ of possible translation block sequences that are searchable by the decoder. Each translation block sequence $\mathbf{z} \in V(\mathbf{S}_i)$ induces a translation, which is then assigned a BLEU score $\mathrm{Bl}(\mathbf{z})$ (obtained by comparing against the target translations). The

goal of the training is to find a weight vector w such that for each training sentence $\mathbf{S}_i$, the corresponding decoder outputs $\hat{\mathbf{z}} \in V(\mathbf{S}_i)$ which has the maximum BLEU score among all $\mathbf{z} \in V(\mathbf{S}_i)$ based on Eq. 2. In other words, if $\hat{\mathbf{z}}$ maximizes the scoring function $s_w(\mathbf{z})$, then $\hat{\mathbf{z}}$ also maximizes the BLEU metric.

Based on the description, a simple idea is to learn the BLEU score $\mathrm{Bl}(\mathbf{z})$ for each candidate block sequence $\mathbf{z}$. That is, we would like to estimate w such that $s_w(\mathbf{z}) \approx \mathrm{Bl}(\mathbf{z})$. This can be achieved through least squares regression. It is easy to see that if we can find a weight vector w that approximates $\mathrm{Bl}(\mathbf{z})$, then the decoding-rule in Eq. 2 automatically maximizes the BLEU score. However, it is usually difficult to estimate $\mathrm{Bl}(\mathbf{z})$ reliably based only on a linear combination of the feature vector as in Eq. 1. We note that a good decoder does not necessarily employ a scoring function that approximates the BLEU score. Instead, we only need to make sure that the top-ranked block sequence obtained by the decoder scoring function has a high BLEU score. To formulate this idea, we attempt to find a decoding parameter such that for each sentence $\mathbf{S}$ in the training data, sequences in $V(\mathbf{S})$ with the highest BLEU scores should get $s_w(\mathbf{z})$ scores higher than those with low BLEU scores.

Denote by $V_K(\mathbf{S})$ a set of K block sequences in $V(\mathbf{S})$ with the highest BLEU scores. Our decoded result should lie in this set. We call them the "truth". The set of the remaining sequences is $V(\mathbf{S}) - V_K(\mathbf{S})$, which we shall refer to as the "alternatives". We look for a weight vector w that minimize the following training criterion:

$$\hat{w} = \arg\min_w \left[\frac{1}{N} \sum_{i=1}^{N} \Phi(w, V_K(\mathbf{S}_i), V(\mathbf{S}_i)) + \lambda w^2 \right] \quad (3)$$

$$\Phi(w, V_K, V) = \frac{1}{K} \sum_{\mathbf{z} \in V_K} \max_{\mathbf{z}' \in V - V_K} \psi(w, \mathbf{z}, \mathbf{z}')$$

$$\psi(w, \mathbf{z}, \mathbf{z}') = \phi(s_w(\mathbf{z}), \mathrm{Bl}(\mathbf{z}); s_w(\mathbf{z}'), \mathrm{Bl}(\mathbf{z}')),$$

where ϕ is a non-negative real-valued loss function (whose specific choice is not critical for the purposes of this paper),and $\lambda \geq 0$ is a regularization parameter. In our experiments, results are obtained using the following convex loss

$$\phi(s, b; s', b') = (b - b')(1 - (s - s'))_+^2, \quad (4)$$

where $(z)_+ = \max(0, z)$. We refer to this formulation as 'costMargin' (cost-sensitive margin) method: for each training sentence S the 'costMargin' $\Phi(w, V_K(\mathbf{S}), V(\mathbf{S}))$ between the 'true' block sequence set $V_K(\mathbf{S})$ and the 'alternative' block sequence set $V(\mathbf{S})$ is maximized. Note that due to the truth and alternative set up, we always have $b > b'$. This loss function gives an upper bound of the error we will suffer if the order of s and s' is wrongly predicted (that is, if we predict $s \leq s'$ instead of $s > s'$). It also has the property that if for the BLEU scores $b \approx b'$ holds, then the loss value is small (proportional to $b - b'$).

A major contribution of this work is a procedure to solve Eq. 3 approximately. The main difficulty is that the search space $V(\mathbf{S})$ covered by the decoder can be extremely large. It cannot be enumerated for practical purposes. Our idea is to replace this large space by a small subspace $V^{(r)}(\mathbf{S}) \subset V(\mathbf{S})$ which we call *relevant set*. The possibility of this reduction is based on the following theoretical result.

Lemma 1 *Let $\psi(w, \mathbf{z}, \mathbf{z}')$ be a non-negative continuous piece-wise differentiable function of w, and let $\hat{w}$ be a local solution of Eq. 3. Let $\xi_i(w, \mathbf{z}) = \max_{\mathbf{z}' \in V(\mathbf{S}_i) - V_K(\mathbf{S}_i)} \psi(w, \mathbf{z}, \mathbf{z}')$, and define*

$$V^{(r)}(\mathbf{S}_i) = \{\mathbf{z}' \in V(\mathbf{S}_i) : \exists \mathbf{z} \in V_K(\mathbf{S}_i) \quad s.t. \quad \xi_i(\hat{w}, \mathbf{z}) \neq 0 \ \& \ \psi(\hat{w}, \mathbf{z}, \mathbf{z}') = \xi_i(\hat{w}, \mathbf{z})\}.$$

Then $\hat{w}$ is a local solution of

$$\min_w \left[\frac{1}{N} \sum_{i=1}^{N} \Phi(w, V_K(\mathbf{S}_i), V^{(r)}(\mathbf{S}_i)) + \lambda w^2 \right]. \quad (5)$$

If ϕ is a convex function of w (as in our choice), then we know that the global optimal solution remains the same if the whole decoding space V is replaced by the relevant set $V^{(r)}$.

Each subspace $V^{(r)}(\mathbf{S}_i)$ will be significantly smaller than $V(\mathbf{S}_i)$. This is because it only includes those alternatives $\mathbf{z}'$ with score $s_{\hat{w}}(\mathbf{z}')$ close to one of the selected truth. These are the most important alternatives that are easily confused with the truth. Essentially the lemma says that if the decoder works well on these difficult alternatives (relevant points), then it works well on the whole space. The idea is closely related to active learning in standard classification problems, where we selectively pick the most important samples (often

Table 1: Generic Approximate Relevant Set Method

for each data point $\mathbf{S}$

initialize truth $V_K(\mathbf{S})$ and alternative $V^{(r)}(\mathbf{S})$

for each decoding iteration l**:** $\ell = 1, \cdots, L$

for each data point $\mathbf{S}$

select relevant points $\{\tilde{\mathbf{z}}_k\} \in V(\mathbf{S})$ (*)

update $V^{(r)}(\mathbf{S}) \leftarrow V^{(r)}(\mathbf{S}) \cup \{\tilde{\mathbf{z}}_k\}$

update w by solving Eq. 5 approximately (**)

based on estimation uncertainty) for labeling in order to maximize classification performance (Lewis and Catlett, 1994). In the active learning setting, as long as we do well on the actively selected samples, we do well on the whole sample space. In our case, as long as we do well on the relevant set, the decoder will perform well.

Since the relevant set depends on the decoder parameter w, and the decoder parameter is optimized on the relevant set, it is necessary to estimate them jointly using an iterative algorithm. The basic idea is to start with a decoding parameter w, and estimate the corresponding relevant set; we then update w based on the relevant set, and iterate this process. The procedure is outlined in Table 1. We intentionally leave the implementation details of the (*) step and (**) step open. Moreover, in this general algorithm, we do not have to assume that $s_w(\mathbf{z})$ has the form of Eq. 1.

A natural question concerning the procedure is its convergence behavior. It can be shown that under mild assumptions, if we pick in (*) an alternative $\tilde{\mathbf{z}}_k \in V(\mathbf{S}) - V_K(\mathbf{S})$ for each $\mathbf{z}_k \in V_K(\mathbf{S})$ $(k = 1, \ldots, K)$ such that

$$\psi(w, \mathbf{z}_k, \tilde{\mathbf{z}}_k) = \max_{\mathbf{z}' \in V(\mathbf{S}) - V_K(\mathbf{S})} \psi(w, \mathbf{z}_k, \mathbf{z}'), \quad (6)$$

then the procedure converges to the solution of Eq. 3. Moreover, the rate of convergence depends only on the property of the loss function, and not on the size of $V(\mathbf{S})$. This property is critical as it shows that as long as Eq. 6 can be computed efficiently, then the Approximate Relevant Set algorithm is efficient. Moreover, it gives a bound on the size of an approximate relevant set with a certain accuracy.[5]

[5]Due to the space limitation, we will not include a formal statement here. A detailed theoretical investigation of the method will be given in a journal paper.

The approximate solution of Eq. 5 in (**) can be implemented using stochastic gradient descent (SGD), where we may simply update w as:

$$w \rightarrow w - \eta \nabla_w \psi(w, \mathbf{z}_k, \tilde{\mathbf{z}}_k).$$

The parameter $\eta > 0$ is a fixed constant often referred to as learning rate. Again, convergence results can be proved for this procedure. Due to the space limitation, we skip the formal statement as well as the corresponding analysis.

Up to this point, we have not assumed any specific form of the decoder scoring function in our algorithm. Now consider Eq. 1 used in our model. We may express it as:

$$s_w(\mathbf{z}) = w^T \cdot F(\mathbf{z}),$$

where $F(\mathbf{z}) = \sum_{i=1}^{n} f(b_i, o_i, b_{i-1})$. Using this feature representation and the loss function in Eq. 4, we obtain the following costMargin SGD update rule for each training data point and k:

$$w \rightarrow w + \eta \Delta \mathrm{Bl}_k x_k \, (1 - w^T \cdot x_k)_+, \quad (7)$$
$$\Delta \mathrm{Bl}_k = \mathrm{Bl}(\mathbf{z}_k) - \mathrm{Bl}(\tilde{\mathbf{z}}_k), \; x_k = F(\mathbf{z}_k) - F(\tilde{\mathbf{z}}_k).$$

4 Experimental Results

We applied the novel discriminative training approach to a standard Arabic-to-English translation task. The training data comes from UN news sources. Some punctuation tokenization and some number classing are carried out on the English and the Arabic training data. We show translation results in terms of the automatic BLEU evaluation metric (Papineni et al., 2002) on the MT03 Arabic-English DARPA evaluation test set consisting of 663 sentences with 16 278 Arabic words with 4 reference translations. In order to speed up the parameter training the original training data is filtered according to the test set: all the Arabic substrings that occur in the test set are computed and the parallel training data is filtered to include only those training sentence pairs that contain at least one out of these phrases: the resulting pre-filtered training data contains about 230 thousand sentence pairs (5.52 million Arabic words and 6.76 million English words). The block set is generated using a phrase-pair selection algorithm similar to (Koehn et al., 2003; Al-Onaizan et al., 2004), which includes some heuristic filtering to

increase phrase translation accuracy. Blocks that occur only once in the training data are included as well.

4.1 Practical Implementation Details

The training algorithm in Table 2 is adapted from Table 1. The training is carried out by running $L = 30$ times over the parallel training data, each time decoding all the $N = 230\,000$ training sentences and generating a single block translation sequence for each training sentence. The top five block sequences $V_5(\mathbf{S}_i)$ with the highest BLEU score are computed up-front for all training sentence pairs S_i and are stored separately as described in Section 2. The score-based decoding of the $230\,000$ training sentence pairs is carried out in parallel on 25 64-Bit Opteron machines. Here, the monotone decoding is much faster than the decoding with block swapping: the monotone decoding takes less than 0.5 hours and the decoding with swapping takes about an hour. Since the training starts with only the parallel training data and a block set, some initial block sequences have to be generated in order to initialize the global model training: for each input sentence a simple bag of blocks translation is generated. For each input interval that is matched by some block b, a single block is added to the bag-of-blocks translation $\mathbf{z}_0(\mathbf{S})$. The order in which the blocks are generated is ignored. For this block set only block and word identity features are generated, i.e. features of type f_{1000} and f_{1001} in Section 2. This step does not require the use of a decoder. The initial training data contains only a single alternative. The training procedure proceeds by iteratively decoding the training data. After each decoding step, the resulting translation block sequences are stored on disc in binary format. A block sequence generated at decoding step ℓ_1 is used in all subsequent training steps ℓ_2, where $\ell_2 > \ell_1$. The training data after the i-th decoding step is given as $\left[\, V_5(\mathbf{S}_i)\,,\, V^{(r)}(\mathbf{S}_i)\,\right]_{i=1}^{N}$, where the size $|V^{(r)}(\mathbf{S}_i)|$ of the relevant alternative set is $i+1$. Although in order to achieve fast convergence with a theoretical guarantee, we should use Eq. 6 to update the relevant set, in reality, this idea is difficult to implement because it requires a more costly decoding step. Therefore in Table 2, we adopt an approximation, where the relevant set is updated by adding the decoder output at each stage. In this way, we are able to treat the decoding scheme as a black box. One way to approximate Eq. 6 is to generate multiple decoding outputs and pick the most relevant points based on Eq. 6. Since the n-best list generation is computationally costly, only a single block sequence is generated for each training sentence pair, reducing the memory requirements for the training algorithm as well. Although we are not able to rigorously prove fast convergence rate for this approximation, it works well in practice, as Figure 2 shows. Theoretically this is because points achieving large values in Eq. 6 tend to have higher chances to become the top-ranked decoder output as well. The SGD-based on-line training algorithm described in Section 3, is carried out after each decoding step to generate the weight vector w for the subsequent decoding step. Since this training step is carried out on a single machine, it dominates the overall computation time. Since each iteration adds a single relevant alternative to the set $V^{(r)}(\mathbf{S}_i)$, computation time increases with the number of training iterations: the initial model is trained in a few minutes, while training the model after the 30-th iteration takes up to 5 hours for the most complex models.

Table 2: Relevant set method: L = number of decoding iterations, N = number of training sentences.

for each input sentence $\mathbf{S}_i,\ i = 1, \cdots, N$
  initialize truth $V_5(\mathbf{S}_i)$ and alternative $V^{(r)} = \{\mathbf{z}_0(\mathbf{S}_i)\}$
for each iteration l**:** $\ell = 1, \cdots, L$
  train w using SGD on training data $\left[\, V_5(\mathbf{S}_i)\,,\, V^{(r)}(\mathbf{S}_i)\,\right]_{i=1}^{N}$
  for each input sentence $\mathbf{S}_i,\ i = 1, \cdots, N$
    select top-scoring sequence $\tilde{\mathbf{z}}(\mathbf{S}_i)$ and update $V^{(r)}(\mathbf{S}_i) \leftarrow V^{(r)}(\mathbf{S}_i) \cup \{\tilde{\mathbf{z}}(\mathbf{S}_i)\}$

Table 3 presents experimental results in terms of uncased BLEU [6]. Two re-ordering restrictions are tested, i.e. monotone decoding ('MON'), and local block re-ordering where neighbor blocks can be swapped ('SWAP'). The 'SWAP' re-ordering uses the same features as the monotone models plus additional orientation-based and distortion-based features. Different feature sets include

[6] Translation performance in terms of cased BLEU is typically reduced by about 2 %.

Table 3: Translation results in terms of uncased BLEU on the training data (230 000 sentences) and the MT03 test data (670 sentences).

	Re-ordering	Features	train	test
1	'MON'	bleu	0.542	-
2		phrase	0.378	0.256
3		word	0.427	0.341
4		both	0.477	0.359
5	'SWAP'	bleu	0.594	-
6		phrase	0.441	0.295
7		word	0.455	0.359
8		both	0.479	0.363

word-based features, phrase-based features, and the combination of both. For the results with word-based features, the decoder still generates phrase-to-phrase translations, but all the scoring is done on the word level. Line 8 shows a BLEU score of 36.3 for the best performing system which uses all word-based and phrase-based features [7]. Line 1 and line 5 of Table 3 show the training data averaged BLEU score obtained by searching for the highest BLEU scoring block sequence for each training sentence pair as described in Section 2. Allowing local block swapping in this search procedure yields a much improved BLEU score of 0.59. The experimental results show that word-based models significantly outperform phrase-based models, the combination of word-based and phrase-based features performs better than those features types taken separately. Additionally, swap-based re-ordering slightly improves performance over monotone decoding. For all experiments, the training BLEU score remains significantly lower than the maximum obtainable BLEU score shown in line 1 and line 5. In this respect, there is significant room for improvements in terms of feature functions and alternative set generation. The word-based models perform surprisingly well, i.e. the model in line 7 uses only three feature types: model 1 features like f_{1001} in Section 2, distortion features like f_{1002}, and target language m-gram features up to $m = 3$. Training speed varies depending on the feature types used: for the simplest model shown in line 2 of Table 3, the training takes about 12 hours, for the models using word-based features shown in line 3 and line 7 training takes less than 2 days. Finally, the training for the most complex model in line 8 takes about 4 days.

[7] With a margin of ± 0.014, the differences between the results in line 4, line 7, and line 8 are not statistically significant, but the other result differences are.

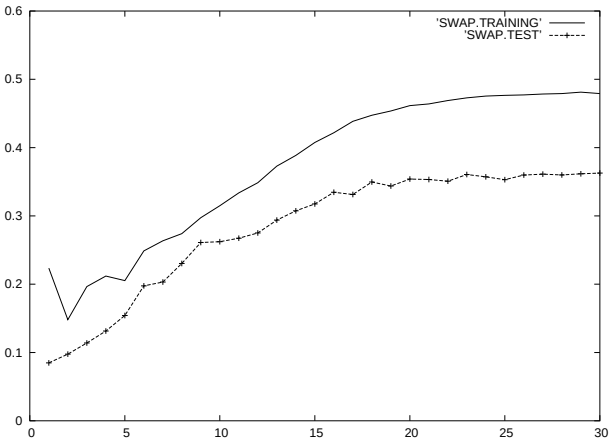

Figure 2: BLEU performance on the training set (upper graph; averaged BLEU with single reference) and the test set (lower graph; BLEU with four references) as a function of the training iteration I for the model corresponding to line 8 in Table 3.

Figure 2 shows the BLEU performance for the model corresponding to line 8 in Table 3 as a function of the number of training iterations. By adding top scoring alternatives in the training algorithm in Table 2, the BLEU performance on the training data improves from about 0.22 for the initial model to about 0.48 for the best model after 30 iterations. After each training iteration the test data is decoded as well. Here, the BLEU performance improves from 0.08 for the initial model to about 0.36 for the final model (we do not include the test data block sequences in the training). Table 3 shows a typical learning curve for the experiments in Table 3: the training BLEU score is much higher than the test set BLEU score despite the fact that the test set uses 4 reference translations.

5 Discussion and Future Work

The work in this paper substantially differs from previous work in SMT based on the noisy channel approach presented in (Brown et al., 1993). While error-driven training techniques are commonly used to improve the performance of phrase-based translation systems (Chiang, 2005; Och, 2003), this paper presents a novel block sequence translation approach to SMT that is similar to sequential natural language annotation problems such as part-of-speech tagging or shallow parsing,

both in modeling and parameter training. Unlike earlier approaches to SMT training, which either rely heavily on domain knowledge, or can only handle a small number of features, this approach treats the decoding process as a black box, and can optimize tens millions of parameters automatically, which makes it applicable to other problems as well. The choice of our formulation is convex, which ensures that we are able to find the global optimal even for large scale problems. The loss function in Eq. 4 may not be optimal, and using different choices may lead to future improvements. Another important direction for performance improvement is to design methods that better approximate Eq. 6. Although at this stage the system performance is not yet better than previous approaches, good translation results are achieved on a standard translation task. While being similar to (Tillmann and Zhang, 2005), the current procedure is more automated with comparable performance. The latter approach requires a decomposition of the decoding scheme into local decision steps with the inherent difficulty acknowledged in (Tillmann and Zhang, 2005). Since such limitation is not present in the current model, improved results may be obtained in the future. A global training procedure in the context of reranking is presented in (Shen et al., 2004).

The computational requirements for the training algorithm in Table 2 can be significantly reduced. While the global training approach presented in this paper is simple, after 15 iterations or so, the alternatives that are being added to the relevant set differ very little from each other, slowing down the training considerably such that the set of possible block translations $V(\mathbf{S})$ might not be fully explored. As mentioned in Section 2, the current approach is still able to handle real-valued features, e.g. the language model probability. This is important since the language model can be trained on a much larger monolingual corpus.

6 Acknowledgment

This work was partially supported by DARPA and monitored by SPAWAR under contract No. N66001-99-2-8916. The author would like to thank the anonymous reviewers for their detailed criticism on this paper.

References

Yaser Al-Onaizan, Niyu Ge, Young-Suk Lee, Kishore Papineni, Fei Xia, and Christoph Tillmann. 2004. IBM Site Report. In *NIST 2004 MT Workshop*, Alexandria, VA, June. IBM.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *CL*, 19(2):263–311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL 2005)*, pages 263–270, Ann Arbor, Michigan, June.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP'02*, Philadelphia,PA.

A. Ittycheriah and S. Roukos. 2005. A Maximum Entropy Word Aligner for Arabic-English MT. In *Proc. of HLT-EMNLP 06*, pages 89–96, Vancouver, British Columbia, Canada, October.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL 2003: Main Proceedings*, pages 127–133, Edmonton, Alberta, Canada, May 27 - June 1.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proc. of HLT-EMNLP 05*, pages 161–168, Vancouver, British Columbia, Canada, October.

D. Lewis and J. Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL'05*, pages 91–98, Ann Arbor, Michigan, June.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL'03*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of machine translation. In *In Proc. of ACL'02*, pages 311–318, Philadelphia, PA, July.

Libin Shen, Anoop Sarkar, and Franz-Josef Och. 2004. Discriminative Reranking of Machine Translation. In *Proceedings of the Joint HLT and NAACL Conference (HLT 04)*, pages 177–184, Boston, MA, May.

Christoph Tillmann and Tong Zhang. 2005. A localized prediction model for statistical machine translation. In *Proceedings of ACL'05*, pages 557–564, Ann Arbor, Michigan, June.